Санкт-Петербургский государственный университет

Кафедра математической теории игр и статистических решений

Винничек Никита Николаевич

Выпускная квалификационная работа бакалавра

«Обслуживание в системе с вирусными заявками»

Направление 010400

Прикладная математика и информатика

Научный руководитель,

кандидат физ.-мат. наук,

доцент, ст. преп.

Домановская Е. Ф.

Санкт-Петербург

2016

Оглавление

Введение

Теория массового обслуживания появилась и стала развиваться относительно недавно. В начале XX столетия был сформулирован ряд математических задач, связанный с практическими требованиями телефонного дела и рациональной организации массового обслуживания. Оказалось, что задачи подобного типа возникают в самых различных направлениях исследований: в экономике, в транспортном деле, в технике, в военно-промышленном деле и в организации производства.

Требования практики выдвигают перед теорией массового обслуживания большое число новых постановок задач. Их рассмотрение необходимо для приложений, для постепенного приближения условий, в которых они решаются, к истинной картине изучаемых явлений. Так, например, ранее не были рассмотрены случаи систем массового обслуживания с вирусными заявками. Входящий поток может состоять не только из реальных запросов на обслуживание, но также и из вредоносных запросов, выводящих обслуживающие каналы или всю систему из строя.

Задачи, в которых следует принимать в расчет вероятность появления вирусных заявок, встречаются постоянно: ложные запросы в курьерской службе доставки; вирусные программы, частично выводящие из строя оперативную память компьютера и т.д.

Можно указать множество других постановок задач реального содержания, которые в своей математической части сводятся к вопросам теории массового обслуживания, включающие в себя вероятность появления вирусных заявок. В данной работе рассматривается именно такой класс задач.

Основной задачей теории массового обслуживания является изучение и исследование режима функционирования обслуживающей системы. Рассмотрим работу системы массового обслуживания (СМО) с вирусными заявками. Исследуем аналитическую модель для одно- и двухканальной СМО и перейдём к её программной интерпретации. На основании проведенных исследований предложим рекомендации по оптимизации работы системы.

Постановка задачи

На n-канальную систему массового обслуживания поступает пуассоновский поток заявок, образованный подпотоками реальных запросов на обслуживание и вирусных заявок с параметрами $\lambda, a\lambda, (1-a)\lambda$ соответственно, $a > 0,5$. Обслуживание реальных заявок экспоненциально с параметром μ. Поступившая вирусная заявка блокирует канал, выводя его из обслуживания, занимает его бесконечно долго при включенной системе. Отказ системы наступает с первой потерей реальной заявки, пришедшей на выключенную систему, это момент обнаружения отказа. Сразу начинается восстановление выведенных из строя каналов обслуживания по экспоненциальному закону с параметром ν.

Можно выделить два режима работы системы:

- Система включена, если система свободна или обслуживаются реальные заявки.
- Система выключена, если пришла вирусная заявка.

Обзор литературы.

Различные элементы теории массового обслуживания рассматриваются во многих книгах и научных изданиях. В первую очередь стоит упомянуть А.Я. Хинчина [8]. Его книги выделяются серьёзностью и глубиной математического анализа. Из российской литература по теории массового обслуживания отметим учебные пособия Г.И. Ивченко, В.А. Каштанова и И.Н. Коваленко [6] и Б.В. Гнеденко, И.Н. Коваленко [3]. Развитию приложений теории массового обслуживания сильно поспособствовала изложение её основ в книге Е.С. Вентцель «Теория вероятностей» [2], написанной простым научно-популярным языком.

Как материал для начального математического курса можно использовать книгу Дж. Кемени и Дж. Снелла «Конечные цепи Маркова» [7], с максимально понятными и элементарно проведенными доказательствами основных теорем.

В теории массового обслуживания за последние десятилетия стали появляться аналитические и общие вероятностные методы, которые позволяют делать о исследуемых системах выводы собирательного характера. Примером таких исследований может служить монографии А.А. Боровкова [1], которые сочетают в себе многообразие вероятностных подходов и интерпретаций с глубокими аналитическими результатами.

Использование теории массового обслуживания на практике невозможно без должной степени развития численных методов. Применяются такие математические системы как Maple и MATLAB для решения задач теории массового обслуживания. Для их освоения и применения на практике существует множество научных пособий: В.П. Дьяконова [5], В.И. Горбаченко [4] и другие.

Глава 1. Аналитическая модель системы

Опишем работу ремонтируемой системы с n обслуживающими каналами, на которые приходят реальные и вирусные заявки. Изобразим графы состояний с интенсивностями переходов. Отметим марковость процесса смены состояний. Вычислим p_{ij} - вероятности переходов из состояния i в состояние j. Составим матрицу переходов $P = (p_{ij})$. Выпишем уравнения Колмогорова для $P_i'(t)$ и найдем показатели эффективности исследуемой модели системы массового обслуживания.

1.1 Описание работы ремонтируемой системы с одним обслуживающим каналом

Рассматриваем случай, когда $n = 1$. $E_i = (b, r)$ - состояния системы, где

b - число вирусных заявок, r - число реальных заявок. Тогда граф состояний с интенсивностями переходов будет выглядеть следующим образом:

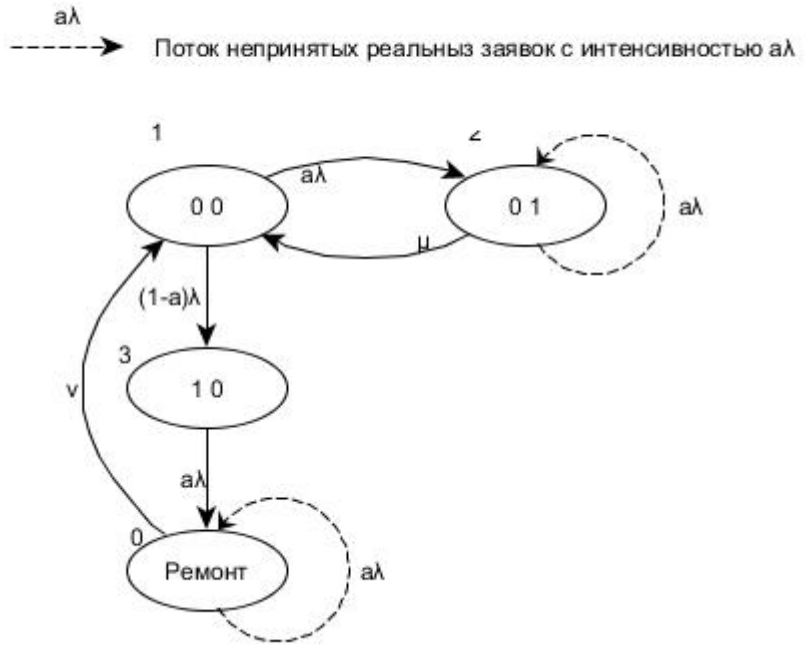

Рис.1

В данном случае (см. рис.1) система может находиться в четырёх состояниях:

- $E_1 = (0,0)$ - простой системы
- $E_2 = (0,1)$ - обработка реальной заявки
- $E_3 = (1,0)$ - система находится в отказе, занята вирусной заявкой
- E_0 - ремонт системы

Найдём вероятности переходов по состояниям. Вычислим вероятность перехода из E_1 в E_2. Вероятность того, что система из E_1 раньше перейдёт в E_2, чем в E_3.

$$P_{12} = P\left((E_1 \to E_2) < (E_1 \to E_3)\right)$$

$$\begin{cases} (E_1 \to E_2) - (\xi, F, f) \\ (E_1 \to E_3) - (\eta, G, g) \\ F(t) = 1 - e^{-a\lambda t} \\ f(t) = a\lambda e^{-a\lambda t} \\ G(t) = 1 - e^{-(1-a)\lambda t} \\ g(t) = (1-a)\lambda e^{-(1-a)\lambda t} \\ \xi, \eta \in (0; +\infty) \end{cases}$$

$$P(\xi < \eta) = P\left(\eta \in (t : t + dt); \xi < t; t \in [0; +\infty)\right) = \int_0^{+\infty} F(t) g(t) dt$$

$$P_{12} = \int_0^{+\infty} (1 - e^{-a\lambda t})(1-a)\lambda e^{-(1-a)\lambda t} = a$$

Тогда вероятность перехода из E_1 в E_3 равняется:

$$P_{13} = 1 - P_{12} = 1 - a$$

Из E_2 возможен переход только в E_1, из E_3 только в E_0, из E_0 только в E_1:

$$\begin{cases} P_{21}=1 \\ P_{30}=1 \\ P_{01}=1 \end{cases}$$

В момент смены состояний переходные вероятности P_{ij} зависят только от номеров состояний. Т.е. эти моменты обладают марковским свойством. Значит, процесс содержит вложенную конечную цепь Маркова.

Матрица переходов:

$$P = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{cccc} 0 & 1 & 2 & 3 \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & a & 1-a \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \end{array}$$

При $N = 6 \Rightarrow$ P^N - транзитивна (все элементы положительны)$\Rightarrow$ по теореме Маркова существует стационарный режим вложенной конечной цепи $\Rightarrow$ Марковской цепи с непрерывным временем.

Теперь составим систему уравнений Чепмена-Колмогорова рассматриваемой системы.

$$P_i(t+\Delta t) = \sum_k P_k(t) P_{ki}(\Delta t); i = \overline{0,3}$$

$$\begin{cases} P_0(t+\Delta t) = P_0(t)P_{00}(\Delta t) + P_3(t)P_{30}(\Delta t) \\ P_1(t+\Delta t) = P_0(t)P_{01}(\Delta t) + P_1(t)P_{11}(\Delta t) + P_2(t)P_{21}(\Delta t) \\ P_2(t+\Delta t) = P_1(t)P_{12}(\Delta t) + P_2(t)P_{22}(\Delta t) \\ P_3(t+\Delta t) = P_1(t)P_{13}(\Delta t) + P_3(t)P_{33}(\Delta t) \end{cases}$$

$$\begin{cases} P_0(t+\Delta t) = P_0(t)(1-\nu\Delta t) + P_3(t)a\lambda\Delta t + o(\Delta t) \\ P_1(t+\Delta t) = P_0(t)\nu\Delta t + P_1(t)(1-\lambda\Delta t) + P_2(t)\mu\Delta t + o(\Delta t) \\ P_2(t+\Delta t) = P_1(t)a\lambda\Delta t + P_2(t)(1-\mu\Delta t) + o(\Delta t) \\ P_3(t+\Delta t) = P_1(t)(1-a)\lambda\Delta t + P_3(t)(1-a\lambda\Delta t) + o(\Delta t) \end{cases}$$

Оставим в правой части уравнений все элементы, содержащие Δt, а в левую перенесем все элементы без Δt:

$$\begin{cases} P_0(t+\Delta t) - P_0(t) = -P_0(t)\nu\Delta t + P_3(t)a\lambda\Delta t + o(\Delta t) \\ P_1(t+\Delta t) - P_1(t) = P_0(t)\nu\Delta t - P_1(t)\lambda\Delta t + P_2(t)\mu\Delta t + o(\Delta t) \\ P_2(t+\Delta t) - P_2(t) = P_1(t)a\lambda\Delta t - P_2(t)\mu\Delta t + o(\Delta t) \\ P_3(t+\Delta t) - P_3(t) = P_1(t)(1-a)\lambda\Delta t - P_3(t)a\lambda\Delta t + o(\Delta t) \end{cases}$$

Теперь разделим обе части на Δt

$$\begin{cases} \dfrac{P_0(t+\Delta t) - P_0(t)}{\Delta t} = -P_0(t)\nu + P_3(t)a\lambda \\ \dfrac{P_1(t+\Delta t) - P_1(t)}{\Delta t} = P_0(t)\nu - P_1(t)\lambda + P_2(t)\mu \\ \dfrac{P_2(t+\Delta t) - P_2(t)}{\Delta t} = P_1(t)a\lambda - P_2(t)\mu \\ \dfrac{P_3(t+\Delta t) - P_3(t)}{\Delta t} = P_1(t)(1-a)\lambda - P_3(t)a\lambda \end{cases}$$

При $\Delta t \to 0$, пределы правых частей уравнений существуют, в силу равенства, они существуют и слева, а тогда это производные функций $P_i(t); i = \overline{0,3}$. Таким образом запишем систему дифференциальных уравнений:

$$\begin{cases} P_0'(t) = -P_0(t)\nu + P_3(t)a\lambda \\ P_1'(t) = P_0(t)\nu - P_1(t)\lambda + P_2(t)\mu \\ P_2'(t) = P_1(t)a\lambda - P_2(t)\mu \\ P_3'(t) = P_1(t)(1-a)\lambda - P_3(t)a\lambda \end{cases}$$

Это система четырех линейных дифференциальных уравнений первого порядка с постоянными коэффициентами. Чтобы решить уравнения Колмогорова и найти вероятности состояний, прежде всего надо задать начальные условия.

$$P_1(0) = 1; P_0(0) = P_2(0) = P_3(0) = 0$$

Можно решить её с помощью пакета прикладных программ MATLAB. Для решения дифференциальных уравнений в форме Коши MATLAB имеет функцию dsolve(‘eqn1’,’eqn2’, …), которая возвращает аналитическое решение системы дифференциальных уравнений с начальными условиями (см. приложение №1). Получаем решение системы дифференциальных уравнений в виде:

$$P_0(t) = P_0(a,\lambda,\mu,\nu,t); P_1(t) = P_1(a,\lambda,\mu,\nu,t);$$
$$P_2(t) = P_2(a,\lambda,\mu,\nu,t); P_3(t) = P_3(a,\lambda,\mu,\nu,t);$$

Найдем финальные вероятности

$$\lim_{t\to\infty} P_i(t) = P_i; i = \overline{0,3};\ \sum_{i=0}^{3} P_i = 1$$

Правые части уравнений системы становятся постоянными, а значит и левые части, в силу равенства, становятся равны $const$. Эти $const$ могут быть только нулями, иначе $\lim_{t\to\infty} P_i(t)$ выйдут из области допустимых значений $[0,1]$, что невозможно для вероятностной меры. Значит, для нахождения финальных вероятностей приравняем все левые части в уравнениях Чепмена-Колмогорова к нулю и решим полученную систему уже не дифференциальных, а линейных алгебраических уравнений:

$$\begin{cases} P_0\nu = P_3 a\lambda \\ P_1\lambda = P_0\nu + P_2\mu \\ P_2\mu = P_1 a\lambda \\ P_3 a\lambda = P_1(1-a)\lambda \end{cases}$$

Для решения линейных алгебраических уравнений MATLAB имеет функцию solve(‘eqn1’,’eqn2’, …), которая возвращает аналитическое решение системы линейных уравнений(см. приложение №1). Получаем решение системы линейных алгебраических уравнений в виде:

$$P_0 = P_0(a,\lambda,\mu,\nu); P_1 = P_1(a,\lambda,\mu,\nu);$$
$$P_2 = P_2(a,\lambda,\mu,\nu); P_3 = P_3(a,\lambda,\mu,\nu);$$

Также можем аналитически решить эту систему, выразив каждое $P_i; i = 0,2,3$ через P_1 :

$$P_2 = \frac{a\lambda}{\mu} P_1; P_3 = \frac{1-a}{a} P_1; P_0 = \frac{1-a}{\nu} P_1 \quad (1.1)$$

$$P_1 = \frac{a\mu\nu}{a^2\lambda\nu + \mu\nu + a\mu - a^2\mu} \quad (1.2)$$

В стационарном режиме рассматриваемая система находится в состояниях E_0, E_1, E_2, E_3 в среднем долю времени равную P_0, P_1, P_2, P_3 соответственно, на промежутке времени, достаточно удаленном от начала функционирования.

Найдем показатели эффективности исследуемой модели системы массового обслуживания:

$$Q = P_1 \quad (1.3)$$

- относительная пропускная способность, заявка может быть принята на обслуживание, система свободна.

$$A = Qa\lambda \quad (1.4)$$

– абсолютная пропускная способность (число обслуженных заявок за ед. времени).

$$P_{\text{потерь}} = P_0 + P_2 + P_3 = 1 - P_1 \quad (1.5)$$

- вероятность возможной потери реальной заявки, система несвободна.

$$B = a\lambda P_{\text{потерь}} = a\lambda(1 - P_1) = a\lambda - A \quad (1.6)$$

- число реальных заявок, получивших отказ за ед. времени, это интенсивность потока потерянных заявок.

1.2 Описание работы ремонтируемой системы с двумя обслуживающими каналами

Рассматриваем случай, когда $n=2$. $E_i=(l,b,r)$ - состояния системы, где l – число ремонтируемых каналов, b - число вирусных заявок, r - число реальных заявок. Ремонт одного или двух каналов начинается при потере реальной заявки, в случае когда система занята двумя вирусными заявками или одной вирусной, одной реальной, происходит обнаружение неисправности системы. Тогда граф состояний с интенсивностями переходов будет выглядеть следующим образом:

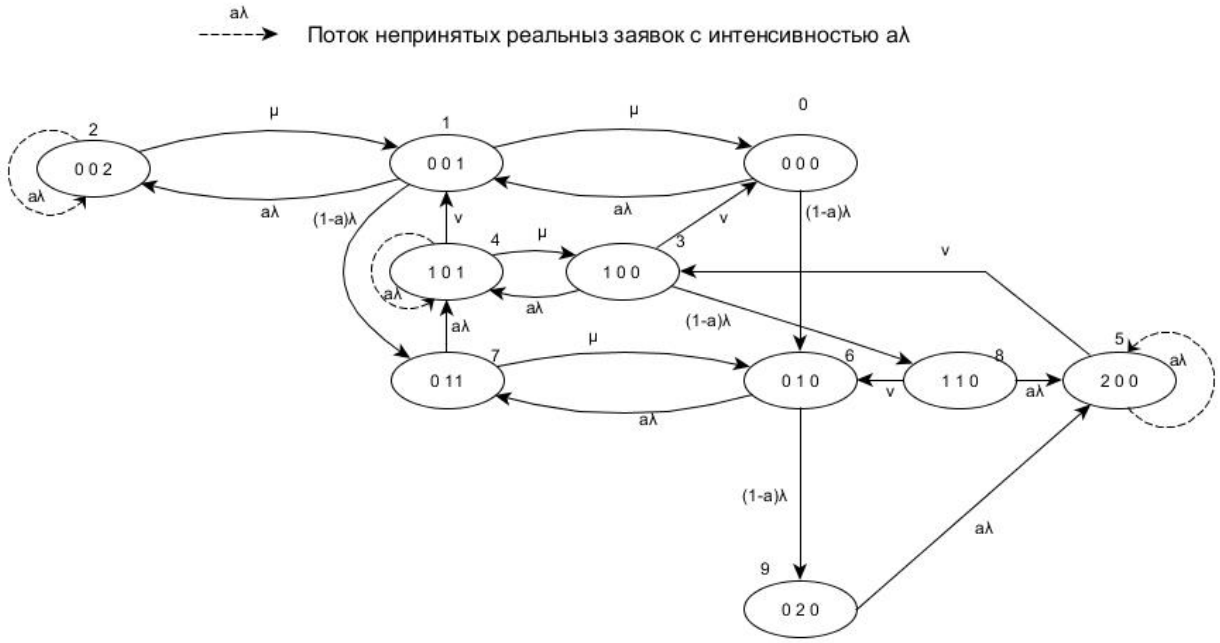

Рис.2

Найдём вероятности переходов между состояниями, изображенными на рис. 2. Матрица переходов:

$$P = \begin{array}{c} \\ 0- \\ 1- \\ 2- \\ 3- \\ 4- \\ 5- \\ 6- \\ 7- \\ 8- \\ 9- \end{array} \begin{array}{c} \begin{array}{cccccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \end{array} \\ \begin{pmatrix} 0 & a & 0 & 0 & 0 & 0 & 1-a & 0 & 0 & 0 \\ \frac{\mu}{\mu+\lambda} & 0 & \frac{a\lambda}{\mu+\lambda} & 0 & 0 & 0 & 0 & \frac{(1-a)\lambda}{\mu+\lambda} & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{\nu}{\nu+\lambda} & 0 & 0 & 0 & \frac{a\lambda}{\nu+\lambda} & 0 & 0 & 0 & \frac{(1-a)\lambda}{\nu+\lambda} & 0 \\ 0 & \frac{\nu}{\nu+\mu} & 0 & \frac{\mu}{\nu+\mu} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & a & 0 & 1-a \\ 0 & 0 & 0 & 0 & \frac{a\lambda}{\mu+a\lambda} & 0 & \frac{\mu}{\mu+a\lambda} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{a\lambda}{\nu+a\lambda} & \frac{\nu}{\nu+a\lambda} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array}$$

При $N = 8 \Rightarrow \quad P^N$ - транзитивна (все элементы положительны) $\Rightarrow$ существует стационарный режим цепи с дискретным временем $\Rightarrow$ существует режим и с непрерывным временем.

Теперь составим систему уравнений Чепмена-Колмогорова рассматриваемой системы. Рассуждаем аналогичным образом, как в случае одноканальной системой:

$$\begin{cases} P_0'(t) = -\lambda P_0(t) + \mu P_1(t) + \nu P_3(t) \\ P_1'(t) = a\lambda P_0(t) - (\lambda + \mu) P_1(t) + \mu P_2(t) + \nu P_4(t) \\ P_2'(t) = a\lambda P_1(t) - \mu P_2(t) \\ P_3'(t) = -(\lambda + \nu) P_3(t) + \mu P_4(t) + \nu P_5(t) \\ P_4'(t) = a\lambda P_3(t) - (\nu + \mu) P_4(t) + a\lambda P_7(t) \\ P_5'(t) = -\nu P_5(t) + a\lambda P_8(t) + a\lambda P_9(t) \\ P_6'(t) = (1-a)\lambda P_0(t) - \lambda P_6(t) + \mu P_7(t) + \nu P_8(t) \\ P_7'(t) = (1-a)\lambda P_1(t) + a\lambda P_6(t) - (a\lambda + \mu) P_7(t) \\ P_8'(t) = (1-a)\lambda P_3(t) - (\nu + a\lambda) P_8(t) \\ P_9'(t) = (1-a)\lambda P_6(t) - a\lambda P_9(t) \\ \sum_{i=0}^{9} P_i(t) = 1 \end{cases}$$

Начальные условия:

$$P_0(0) = 1; P_i(0) = 0, i = \overline{1,9}$$

Решаем её с помощью функции dsolve() пакета прикладных программ MATLAB (см. приложение №2). Получаем решение системы дифференциальных уравнений в виде:

$$P_0(t) = P_0(a,\lambda,\mu,\nu,t); P_1(t) = P_1(a,\lambda,\mu,\nu,t); P_2(t) = P_2(a,\lambda,\mu,\nu,t); P_3(t) = P_3(a,\lambda,\mu,\nu,t);$$

$$P_4(t) = P_4(a,\lambda,\mu,\nu,t); P_5(t) = P_5(a,\lambda,\mu,\nu,t); P_6(t) = P_6(a,\lambda,\mu,\nu,t); P_7(t) = P_7(a,\lambda,\mu,\nu,t);$$

$$P_8(t) = P_8(a,\lambda,\mu,\nu,t); P_9(t) = P_9(a,\lambda,\mu,\nu,t);$$

Для стационарного режима система выглядит следующим образом:

$$\begin{cases} \lambda P_0 = \mu P_1 + \nu P_3 \\ (\lambda + \mu) P_1 = a\lambda P_0 + \mu P_2 + \nu P_4 \\ \mu P_2 = a\lambda P_1 \\ (\lambda + \nu) P_3 = \mu P_4 + \nu P_5 \\ (\nu + \mu) P_4 = a\lambda P_3 + a\lambda P_7 \\ \nu P_5 = a\lambda P_8 + a\lambda P_9(t) \\ \lambda P_6 = (1-a)\lambda P_0 + \mu P_7 + \nu P_8 \\ (a\lambda + \mu) P_7 = (1-a)\lambda P_1 + a\lambda P_6 \\ (\nu + a\lambda) P_8 = (1-a)\lambda P_3 \\ a\lambda P_9 = (1-a)\lambda P_6 \\ \sum_{i=0}^{9} P_i = 1 \end{cases}$$

Решаем её с помощью функции solve() пакета прикладных программ MATLAB (см. приложение №2). Получаем решение системы линейных алгебраических уравнений в виде:

$$P_0 = P_0(a,\lambda,\mu,\nu); P_1 = P_1(a,\lambda,\mu,\nu); P_2 = P_2(a,\lambda,\mu,\nu); P_3 = P_3(a,\lambda,\mu,\nu);$$

$$P_4 = P_4(a,\lambda,\mu,\nu); P_5 = P_5(a,\lambda,\mu,\nu); P_6 = P_6(a,\lambda,\mu,\nu); P_7 = P_7(a,\lambda,\mu,\nu);$$

$$P_8 = P_8(a,\lambda,\mu,\nu); P_9 = P_9(a,\lambda,\mu,\nu);$$

Также можем аналитически решить эту систему, выразив каждое P_i через P_0 и постоянные C и D:

$$C = a^2\lambda^3 - a^2\lambda\ \mu^2 - a^2\lambda\ \nu\ \mu\ + a^2\lambda\ \nu^2 + a\ \lambda^2\mu + 2a\ \lambda^2\nu\ + a\ \lambda\ \mu^2 + a\ \lambda\ \nu\ \mu\ - a\ \lambda\ \nu^2 - a\ \nu\ \mu^2 - a\ \nu^2\mu\ + \lambda\ \nu\ \mu\ + \lambda\nu^2 + \nu\ \mu^2 + \nu^2\mu$$

$$D = a^4\lambda^3\nu + a^3\lambda^4 + a^3\lambda^3\nu + a^3\lambda^2\mu - a^3\lambda^2\mu^2 - a^3\lambda^2\nu\mu - a^4\lambda^3\nu^2 + a^2\lambda^3\mu + 2a^2\lambda^3\nu + a^2\lambda^2\mu^2 + a^2\lambda^2\nu\mu + 2a^2\lambda^2\nu^2 - a^2\lambda\ \nu^2\mu^2 - a^2\lambda\ \nu^2\mu\ + a\ \lambda^2\nu\ \mu\ + a\ \lambda\ \nu\ \mu^2 + a\ \lambda\ \nu^2\mu$$

$$P_1 = \frac{D}{\mu C} P_0\ ; P_2 = \frac{a\lambda D}{\mu^2 C} P_0\ ;\ P_3 = \frac{(\lambda C - D)}{\nu C} P_0\ ; P_4 = \frac{-a\lambda\mu C + (\lambda + \mu - a\lambda) D}{\mu\nu C} P_0\ ;$$

$$P_5 = \frac{(\lambda + \nu - a\mu)\lambda C + (\mu - a\lambda - \nu)D}{\nu^2 C} P_0;$$

$$P_6 = \left(\frac{(a-1)D}{a\mu C} + \frac{a\lambda + \mu}{a\lambda} \left(-\frac{\lambda + \nu + \mu}{\nu} + \frac{(\nu\lambda + \mu\lambda + \mu\nu + \mu^2 - a\nu\lambda)D}{a\mu\nu C} \right) \right) P_0;$$

$$P_7 = \left(-\frac{\lambda + \nu + \mu}{\nu} + \frac{(\nu\lambda + \mu\lambda + \mu\nu + \mu^2 - a\nu\lambda)D}{a\mu\nu C} \right) P_0 \; ; P_8 = \frac{(1-a)\lambda(\lambda C - D)}{(\nu + a\lambda)\nu C} P_0 \; ;$$

$$P_9 = \left(\frac{1-a}{a} \right) \left(\frac{(a-1)D}{a\mu C} + \frac{a\lambda + \mu}{a\lambda} \left(-\frac{\lambda + \nu + \mu}{\nu} + \frac{(\nu\lambda + \mu\lambda + \mu\nu + \mu^2 - a\nu\lambda)D}{a\mu\nu C} \right) \right) P_0;$$

$$P_0 = \left(1 + \frac{1}{P_0} \sum_{i=1}^{9} P_i \right)^{-1};$$

Найдем показатели эффективности исследуемой двухканальной модели системы массового обслуживания:

$Q = P_0 + P_1 + P_3 + P_6$ - относительная пропускная способность, заявка может быть принята на обслуживание, система полностью свободна или есть один свободный канал.

$A = Qa\lambda$ – абсолютная пропускная способность (число обслуженных реальных заявок за ед. времени).

$P_{\text{потери}} = 1 - Q$ - вероятность потери реальной заявки.

$N_{\text{потерь}} = a\lambda P_{\text{потери}}$- число реальных заявок, получивших отказ за ед. времени, интенсивность потока потерянных реальных заявок.

1.3 Переход к программной модели системы массового обслуживания

Рассмотрев случаи одно- и двухканальной системы массового обслуживания, отметим отсутствие четкой закономерности построения графа

переходов между состояниями системы, а соответственно сложность в построении или отсутствие общей аналитической модели для случая n-канальной системы.

Для изучения работы рассматриваемой системы в общем виде напишем программу, моделирующую систему массового обслуживания с вирусными заявками и с возможностью ремонта. По введенным параметрам: интенсивность входящих потоков вирусных и реальных заявок, интенсивность обслуживания реальных заявок, интенсивность ремонта выведенной из строя системы - программа должна смоделировать процесс работы изучаемой системы массового обслуживания. Тогда для конкретных параметров мы сможем определить оптимальное число обслуживающих каналов. Так, одной из характеристик обслуживающей системы является среднее число потерянных реальных заявок за единицу времени, которая уменьшается при увеличении числа обслуживающих каналов. Однако каждый дополнительный канал требует материальных затрат, что является негативным фактором. Следовательно, в теории возникают задачи оптимизации: каким образом достичь определенного уровня обслуживания (максимального сокращения числа потерянных заявок) при минимальных затратах, связанных с затратами на обслуживание каналов. Именно для такого анализа и необходимо программа, моделирующая систему массового обслуживания позволяющая провести анализ.

Глава 2. Программная модель рассматриваемой системы

В качестве языка программирования был выбран C++, который удовлетворяет всем требованиям реализации программной модели для общего случая рассматриваемой системы массового обслуживания.

Генерируется поток заявок, подчиняющийся пуассоновскому закону распределения с задаваемым параметром, который поступает на обслуживающие каналы. Интервал времени моделирования работы системы задается в программе. Время обслуживания реальной заявки и ремонта выведенного из строя канала, также определяется случайным образом и подчиняются экспоненциальному закону с задаваемыми параметрами. Время моделирования и время обслуживания заявок генерируется в условных единицах времени.

Когда теряется реальная заявка, происходит проверка исправности обслуживающих каналов. При обнаружении хотя бы одного выведенного из строя канала, начинается ремонт системы, состоящий в ремонте всех неисправных каналов.

Программа фиксирует изменение состояния системы в каждую единицу времени моделирования и отображает полученный результат. По окончании работы программы выводятся доли времени нахождения системы в каждом состоянии, количество обработанных и потерянных реальных заявок, производительность каждого канала (среднее число обработанных реальных заявок за единицу времени) и производительность системы в целом.

2.1 Описание работы программной модели

Основными исполняемыми модулями программы, моделирующей рассматриваемую систему массового обслуживания (см. приложение №3), являются классы Channel и MaintenanceSystem.

Класс Channel ответственен за работу каждого канала и имеет следующие внутренние параметры:

int status; - переменная, отвечающая за текущее состояние данного канала, принимает значения «0» (свободен), «1» (обрабатывает реальную заявку) или «-1» (выведен из строя).

bool status_changed; - логическая переменная, сигнализирующая о том, что канал изменил свое состояние.

int real_app_done; - переменная-счётчик, в которой фиксируется количество обработанных каналом реальных заявок.

bool virus; - логическая переменная, сигнализирующая о том, что канал выведен из строя.

int time_real; - переменная-счётчик, в которой указывается общее время, затраченное на обработку каналом реальных заявок.

int time_free; - переменная-счётчик, в которой указывается общее время простоя канала.

int income_real; - переменная-счётчик, в которой фиксируется число заявок пришедших на канал для обслуживания (в независимости от того, были ли они приняты или нет).

int time_virus; - переменная-счётчик, в которой указывается длительность состояния, когда канал выведен из строя.

В данном классе при поступлении заявки на канал происходят проверка типа данной заявки и соответствующие действия с ней внутри канала.

Класс MaintenanceSystem выполняет работу всей системы массового обслуживания и имеет следующие внутренние параметры:

vector<Channel> channels; - вектор, состоящий из множества классов типа " Channel ".

vector<vector<int> > status_history; - вектор векторов, которые хранят в себе состояния системы в каждый момент времени. Отображает динамику изменения состояний системы.

map<vector<int>, int> status_statistics; - вектор типа map, состоящий из всех возможных состояний, в которых побывала система, и время, проведенное в этих состояниях.

int T; - время работы всей системы.

int num_lost; - число потерянных реальных заявок.

int num_accepted; - число обслуженных реальных заявок.

void run() – функция, описывающая генерацию входящего потока заявок и работу системы.

В данном классе генерируется входящий поток заявок и подается на обслуживающие каналы. В файл "stats.txt" выводится статистика работы системы, включающая в себя все возможные состояния системы и доли времени, которые система находилась в этих состояниях, производительность каждого обслуживающего канала и всей системы в целом, число обслуженных и потерянных заявок. В файл "process.txt" выводится динамика работы системы: последовательный переход по состояниям системы и время, проведенное в этих состояниях.

2.2 Сравнение программной и аналитической моделей

Для того чтобы показать адекватность и надежность программной модели рассматриваемой системы массового обслуживания, сравним её с

аналитической моделью для случаев одно- и двухканальной системы при конкретных значениях параметров системы.

Пусть:

$$a = 0.8; \lambda = 0.0425; \mu = 0.25; \nu = 0.05;$$

Тогда, подставив эти значения в (1.1) - (1.6) и задав такие же значения параметров в программе, получим:

$n = 1$	**Аналитическая модель**	**Программная модель**
A	0.02128	0.02122
$\boldsymbol{P}_{\text{потерь}}$	0.3741	0.370185
$\boldsymbol{N}_{\text{потерь}}$	0.012719	0.0132639
Q	0.6259	0. 629815

Табл.№1

В таблице №1 приведены следующие характеристики системы:

n - количество обслуживающих каналов.

Q - относительная пропускная способность.

A - абсолютная пропускная способность (число обслуженных заявок за ед. времени).

$P_{\text{потерь}}$ - вероятность возможной потери реальной заявки.

$N_{\text{потерь}}$ - число реальных заявок, получивших отказ за ед. времени.

Теперь рассмотри эти характеристики для случая $n = 2$:

$n=2$	**Аналитическая модель**	**Программная модель**
A	0.0282064	0.0279861
$\boldsymbol{P}_{\text{потерь}}$	0.1704	0.170213
$\boldsymbol{N}_{\text{потерь}}$	0.0057936	0.00574
Q	0.8296	0.829787

Табл. №2

Как видно из таблиц №1 и №2 погрешность вычисления составляет тысячные доли, что является очень хорошим показателем. На основании этих данных можно заключить, что программная модель соответствует аналитической, а значит является достоверной и надёжной.

2.3 Пример. Физическая модель ремонтируемой системы массового обслуживания с вирусными заявками

В качестве физической модели описанной выше системы массового обслуживания рассмотрим курьерскую службу доставки еды. Вирусной заявкой будем считать ложный заказ. По статистике каждый пятнадцатый заказ является ложным, когда курьер прибывает на место, указанное заказчиком, ожидает получателя, но в итоге никто так и не забирает заказ. Учитывая, что еда может быть скоропортящейся или просто теряет свой товарный вид через какое-то время, то получатся, что компания терпит убытки от каждого такого заказа. Причин ложных заказов может быть несколько: когда конкуренты, например, начинают названивать, делать заказы на большие суммы и указывать неправильные адреса, человек просто передумал заказывать еду, а отменить свой заказ уже поздно, или просто забыл о сделанном заказе.

Обслуживающим каналом будем называть курьера службы доставки, обслуживание реальной заявки – процесс приготовления еды и доставки, ремонтом – время ожидания заказчика по прибытии к месту доставки и утилизация продукта.

Изменим код нашей программы таким образом, чтобы он удовлетворял условиям поставленной задаче (см. приложение №4).

Пусть в фирму по доставке еды в среднем обращается 600 человек в сутки, каждая пятнадцатая заявка является вирусной, на обслуживание одной заявки уходит в среднем 1 час, среднее время ожидания клиента по прибытии к месту доставки составляет 20 минут. Получаем параметры нашей системы: $\lambda = 0.416$ (заявок за минуту), $(1-a) = 0.07, a = 0.93$, $\mu = 0.017$ (заявок в минуту), $\nu = 0.05$ (заявок в минуту). Регулируя количество обслуживающих каналов можем проследить динамику работы системы:

Время работы системы	1 месяц	1 месяц	1 месяц	1 месяц	1 месяц	1 месяц
Количество обслуживающих каналов	100	50	42	30	2	1
Число поступивших заявок	16628	17024	16905	16755	16830	17540
Число обслуженных заявок	16628	17024	16905	15694	1257	626
Число потерянных заявок	0	0	0	1061	15573	16914
Количество задействованных каналов	41	42	42	30	2	1

Табл. №3

По результатам таблицы №3 можно сделать следующий вывод: оптимальное количество обслуживающих каналов в системе равно 42, в этом случае потери заявок являются незначительными, и нет простаивающих каналов.

Выводы

В процессе подготовки ВКР была исследована ремонтируемая система массового обслуживания с вирусными заявками. Аналитически были описаны следующие модели: одно- и двухканальной СМО, вычислены стационарные и абсолютные вероятности, а также средние значения числа обслуженных и потерянных реальных заявок за единицу времени.

В ходе работы была написана программа, позволяющая моделировать систему массового обслуживания с вирусными заявками с n обслуживающими каналами. Практически в данной работе была реализована функциональная модель, позволяющая прослеживать изменения поведения состояния системы.

Проведен анализ аналитических и компьютерной моделей и выявлено их полное соответствие.

В результате работы была выработана способность практического применения описанной модели к реальным задачам. Программная модель применена для исследования работы курьерской службы доставки еды. На основании проведенного исследования вынесены рекомендации по оптимизации работы системы.

Заключение

Цель настоящей работы заключается в исследовании ремонтируемой системы массового обслуживания с вирусными заявками и возможности применить её в деле. При решении поставленной задачи построены аналитические и программная модели. Их исследование необходимо для приложений, для постепенного приближения условий, в которых они решаются, к задачам, выдвигаемым требованиями практики.

Таким образом, исследован новый класс задач теории массового обслуживания, вероятно, раннее не освящавшийся в литературе по данной тематике. Возможно дальнейшее его изучение и применение к задачам, постановки которых приближены к реальным условиям.

Список литературы

1. Боровков А.А. Вероятностные процессы в теории массового обслуживания. М.: Наука, 1972. 357с.
2. Вентцель Е.С. Теория вероятностей: Учеб. для вузов. 6-е изд. стер. М.: Высш. шк., 1999. 576 с.
3. Гнеденко Б.В., Коваленко И.Н. Введение в теорию массового обслуживания. Изд. 5-е, испр. М.: Издательство ЛКИ, 2011. 400с.
4. Горбаченко В.И. Вычислительная линейная алгебра с примерами на MATLAB. СПб.: БХВ-Петербург, 2011. 318 с.
5. Дьяконов В.П. MATLAB 6.0/6.1/6.5/6.5 + SP1 + Simulink 4/5. Обработка сигналов и изображений. М.: СОЛОН-Пресс, 2004. 592 с.
6. Ивченко Г.И., Каштанов В.А., Коваленко И.Н. Теория массового обслуживания. М.: ЛИБРОКОМ, 2012. 304 с.
7. Кемени Д. Дж., Снелл Дж. Л. Конечные цепи Маркова. М.: Наука, 1970. 271 с.
8. Хинчин А. Я. Работы по математической теории массового обслуживания / Под редакцией Б. В. Гнеденко. М.: Физматгиз, 1963. 236 с.

Приложение

Приложение №1

```
syms n m t a l;
%Где n - ню, m- мю, t- время, l - лямбда
%Решаем систему диффиринциальных уравнений
dsolve('Dx0(t) = -n*x0(t) + a*l*x3(t)','Dx1(t) = -l*x1(t)+n*x0(t)+m*x2(t)','Dx2(t) = -
m*x2(t)+a*l*x1(t)','Dx3(t) = -a*l*x3(t)+(1-a)*l*x1(t)','x0(0)=0','x1(0)=1','x2(0)
=0','x3(0)=0');
 %Решаем систему линейных алгебраических уравнений
 solve ('0 = -n*x0 + a*l*x3','0 = -l*x1+n*x0+m*x2','0 = -m*x2+a*l*x1','x0+x1+x2+x3=1')
```

Приложение №2

```
syms n m t a l;
%Где n - ню, m- мю, t- время, l - лямбда
solve ('Dx0(t)=-l*x0(t)+n*x3(t)+m*x1(t)','Dx1(t)=a*l*x0(t)-(l+m)*x1(t)+m*x2(t)+n*x4
(t)','Dx2(t)=a*l*x1(t)-m*x2(t)','Dx3(t)=-(l+n)*x3(t)+m*x4(t)+n*x5(t)','Dx4(t)=a*l*x3(t)
-(n+m)*x4(t)+a*l*x7(t)','Dx5(t)=-n*x5(t)+a*l*x8(t)+a*l*x9(t)','Dx6(t)=(1-a)*l*x0(t)-
l*x6(t)+m*x7(t)+n*x8(t)','Dx7(t)=(1-a)*l*x1(t)+a*l*x6(t)-(a*l+m)*x7(t)','Dx8(t)=(1-a)
*l*x3(t)-(n+a*l)*x8(t)=0','Dx9(t)=(1-a)*l*x6(t)-a*l*x9(t)=0','x0(0)=1','x1(0)=0','x2(0)
=0','x2(0)=0','x3(0)=0','x4(0)=0','x5(0)=0','x6(0)=0','x7(0)=0','x8(0)=0','x9(0)=0');
%Решаем систему диффиринциальных уравнений
solve ('-l*x0+n*x3+m*x1=0','a*l*x0-(l+m)*x1+m*x2+n*x4=0','a*l*x1-m*x2=0','-(l+n)
*x3+m*x4+n*x5=0','a*l*x3-(n+m)*x4+a*l*x7=0','-n*x5+a*l*x8+a*l*x9=0','(1-a)*l*x0-
l*x6+m*x7+n*x8=0','(1-a)*l*x3-(n+a*l)*x8=0','(1-a)*l*x6-
a*l*x9=0','x0+x1+x2+x3+x4+x5+x6+x7+x8+x9=1');
```

Приложение №3

```
#include<iostream>
#include <random>
#include<vector>
#include<fstream>
#include<time.h>
#include<map>
#include<iterator>
using namespace std;
//Класс "Канал"
class Channel
{
public:
    //переменная отвечающая за текущее состояние
    //данного канала, принимает значения 0 (свободен), 1 (обрабатывает реальную заявку) или -1 (выведен из строя)
    int status;
    //логическая переменная, сигнализирующая о том, что канал изменил свое состояние
    bool status_changed;
    //переменная-счётчик, в которой находится количество обработанных каналом реальных заявок
    int real_app_done;
    //логическая переменная, сигнализирующая о том, что канал выведен из строя
    bool virus;
    //переменная-счётчик, в которой находится количество времени, затраченное на обработку каналом реальных заявок
    int time_real;
    //переменная-счётчик, в которой находится количество времени простоя канала
    int time_free;
    //переменная-счётчик, в которой находится количество заявок,
    //число пришедших заявок на канал для обслуживания (в независимости были ли они приняты или нет)
    int income_real;
    //переменная-счётчик, в которой находится количество времени в состоянии, когда канал выведен из строя
    int time_virus;
    //функция подачи заявки на канал
    bool input(int &real_application,                          //не равна 0 когда в данный момент на канал пришла реальная заявка
        int &virus_application,                                //не равна 0 когда в данный момент на канал пришла реальная заявка
        poisson_distribution<int> &distribution_processing, //встроенные методы иммитации распределения ПУАССОНА для интенсивности обработки реальной заявки
        poisson_distribution<int> &distribution_repair,     //встроенные методы иммитации распределения ПУАССОНА для интенсивности ремонта выведенного из строя канала
        default_random_engine &generator)                      //встроенный генератор случайных чисел
    {
        status_changed = false;
        //если пришла реальная заявка, накпливаем число пришедших на канал реальных заявок
        if (real_application)
            income_real++;
```

```
43          if (!virus)
44          {
45              //если канал занят заявкой
46              if (status != 0)
47              {
48                  //если занят реальной завкой
49                  if (status == 1)
50                  {
51                      //накапливаем время затраченное на обрабатку данной заявки
52                      time_real++;
53                      //канал обрабатывает заявку
54                      //результат зависит от интенсивности обслуживания канала
55                      //в зависимости от которой строится распределение Пуассона
56                      int try_to_process = distribution_processing(generator);
57                      //проверяем, обработли ли мы эту заявку в данный момент
                        времени
58                      if (try_to_process)
59                      {
60                          status = 0;
61                          status_changed = true;
62                      }
63                  }
64                  //если занят вирусной заявкой
65                  else
66                  {
67                      //накапливаем время, которое канал провел в выведенном из
                        строя состоянии
68                      time_virus++;
69                      //канал ремонтироуется
70                      //результат зависит от интенсивности ремонта канала
71                      //в зависимости от которой строится распределение Пуассона
72                      int try_to_repair = distribution_repair(generator);
73                      //проверяем, отремонтировался ли канал в данный момент
                        времени
74                      if (try_to_repair)
75                      {
76                          status = 0;
77                          status_changed = true;
78                      }
79                  }
80              }
81              //если канал не занят, и пришли вирусная или реальная заявка
82              else if (virus_application > 0 || real_application > 0)
83              {
84                  status_changed = true;
85                  //если пришла только одна заявка (либо вирусная, либо
                      реальная)
86                  if (!(virus_application > 0 && real_application > 0))
87                  {
88                      //если это реальная заявка
89                      if (real_application > 0)
90                      {
91                          //меняем соответсвующие характеристки канала
92                          status = 1;
93                          real_app_done++;
```

```
                        real_application--;
                    }
                    //если это вирусная заявка
                    else
                    {
                        //меняяем соответсвующие характеристки канала
                        status = -1;
                        virus_application--;
                        virus = true;
                    }
                    return true;
                }
                //если одновременно пришли вирусная и реальная заявка
                else
                {
                    //случайным образом выбираем, которая из них займет канал
                    int choise = rand() % 2;
                    if (choise == 0)
                    {
                        status = -1;
                        virus_application--;
                        virus = true;
                    }
                    else
                    {
                        status = 1;
                        real_app_done++;
                        real_application--;
                    }
                    return true;
                }
            }
            //если не пришло ни одной заявки
            else
            {
                //накапливаем время простоя канала
                time_free++;
                return false;
            }
        }
        //если канал выведен из строя
        else
        {
            // накапливаем время, когда канал выведен из строя
            time_virus++;
            return false;
        }
    }
    //начальные характеристики системы
    Channel()
    {
        time_real = 0;
        time_free = 0;
        time_virus = 0;
        real_app_done = 0;
        status = 0;
```

```
150         income_real = 0;
151     }
152     ~Channel() {};
153 private:
154 
155 };
156 //класс "Система обслуживания"
157 class MaintenanceSystem
158 {
159 public:
160     vector<Channel> channels;                    //вектор, состоящий из
      множества классов типа "Канал"
161     vector<vector<int> > status_history;         //вектор векторов, которые
      хранят в себе состояния системы в каждый момент времени,
162                                                  //отображает динамику
                    изменения состояний системы
163     map<vector<int>, int> status_statistics;     //вектор, состоящий из всех
      возможных состояний, в которых побывала система,
164                                                  //и количество времени
                    проведенного в этих состояниях
165     //время работы всей системы
166     int T;
167     //число потерянных реальных заявок
168     int num_lost;
169     //число обслуженных реальных заявок
170     int num_accepted;
171     //оределяем сколько в нашей системе обслуживающих каналов
172     MaintenanceSystem(int num_of_channels)
173     {
174         channels.resize(num_of_channels, Channel());
175         num_lost = 0;
176         num_accepted = 0;
177     }
178     //функция описывающая входящий поток заявок и работу системы
179     void run(double real_intens, double virus_intens, double
      processing_intens, double repair_intens, int _T)
180     {
181         T = _T;
182         vector<int> status(channels.size(), 0);
183         status_history.push_back(status);
184         int length = 0;
185         ofstream out("process.txt");
186         ofstream out_s("sequence.txt");
187         default_random_engine generator;
188         poisson_distribution<int> distribution_virus(virus_intens);
                //гененрируется объекты подчиняющиеся распределению
189         poisson_distribution<int> distribution_real(real_intens);
                //ПУАССОНА с заданными интенсивностями
190         poisson_distribution<int> distribution_processing(processing_intens);
191         poisson_distribution<int> distribution_repair(repair_intens);
192         int count_real = 0, count_virus = 0;
193         for (int i = 0; i < T; i++)
194         {
195             //получаем состояние поток заявок в данный момент времени
```

```
196             //в зависимости от интенсивности поступелния заявок
197             int real = distribution_real(generator);
198             int virus = distribution_virus(generator);
199             count_real += real;
200             count_virus += virus;
201             //выводим это состояние потока в файл
202             out_s << real << ' ' << virus << endl;
203             length++;
204             bool status_changed = false;
205             bool accepted = false;
206             //подаём это состояние потока на обслуживающие каналы и проводим с
                    ним соответсвующие операции
207             for (int c = 0; c < channels.size(); c++)
208             {
209                 bool  application_accepted = channels[c].input(real, virus,
                      distribution_processing, distribution_repair, generator);
210                 if (channels[c].status_changed)
211                 {
212                     status_changed = true;
213                     status[c] = channels[c].status;
214                 }
215             }
216             if (status_changed)
217             {
218                 for (int i = 0; i < status_history.back().size(); i++)
219                 {
220                     out.width(3);
221                     out << status_history.back()[i];
222                 }
223                 out.width(7);
224                 out << length << " it = ";
225                 out.width(11);
226                 out << double(length) / T * 100 << " %" << endl;
227                 status_statistics[status_history.back()] += length;
228                 status_history.push_back(status);
229                 length = 0;
230             }
231             if (real > 0)
232             {
233                 for (int c = 0; c < channels.size(); c++)
234                 {
235                     channels[c].virus = false;
236                 }
237             }
238         }
239         status_statistics[status_history.back()] += length;
240         status_history.push_back(status);
241         length = 0;
242         // выводим в файл "stats.txt" статистику работы системы
243         out = ofstream("stats.txt");
244         out << endl << endl << "############## STATS ##############" << endl <<
              endl;
245         map <vector<int>, int>::iterator cur;
246         // выводим состояния системы и доли времени, которые система находится
                в этих состояниях
247         for (cur = status_statistics.begin();cur != status_statistics.end
```

```
            ();cur++)
248         {
249             for (int i = 0; i < (*cur).first.size(); i++)
250             {
251                 out.width(3);
252                 out << (*cur).first[i];
253             }
254             out.width(10);
255             out<< (*cur).second<< " it = ";
256             out.width(7);
257             out << double((*cur).second) / T * 100 << " %" << endl;
258         }
259         // производительность каждого канала
260         out << endl << "real applications for channel:" << endl;
261         double summ = 0;
262         for (int i = 0; i < channels.size(); i++)
263         {
264             out << "channel " << i + 1 << " = " << channels[i].real_app_done
                  << " = " << double(channels[i].real_app_done) / T << " per t"
                  <<"    income real = "<<channels[i].income_real<< endl;
265             summ += double(channels[i].real_app_done) / T;
266             num_accepted += channels[i].real_app_done;
267         }
268         num_lost = count_real - num_accepted;
269         //общая производительность системы
270         out << "summ = " << summ << endl;
271         //число обслуженных и потерянных заявок
272         out << endl << "TOTAL REAL ACCEPTED = " << num_accepted << " = " <<
            double(num_accepted) / (num_lost + num_accepted) * 100 << " %" <<
            endl;
273         out << endl << "TOTAL REAL LOST = " << num_lost << " = " << double
            (num_lost) / (num_lost + num_accepted) * 100 << " %" << "\t" <<
            double(num_lost) / T << endl;
274     }
275     ~MaintenanceSystem() {}
276 private:
277
278 };
279
280 int main()
281 {
282     srand(time(NULL));
283     //Требуется ввести показатели системы и выбрать число обслуживающих
        каналов
284     double lamda = 0.138, a = 0.9;
285     double virus_intens = (1 - a)*lamda, real_intens = a*lamda;
286     double processing_intens = 0.025, repair_intens = 0.05;
287     int T = 200000;
288     MaintenanceSystem sys(8);
289     sys.run(real_intens, virus_intens, processing_intens, repair_intens, T);
290 }
```

Приложение №4

```
#include<iostream>
#include <random>
#include<vector>
#include<fstream>
#include<time.h>
#include<map>
#include<iterator>
using namespace std;
class Channel
{
public:
    int status;
    bool status_changed;
    int real_app_done;
    bool virus;
    int time_real;
    int time_free;
    int income_real;
    int time_virus;
    bool input(int &real_application,
        int &virus_application,
        poisson_distribution<int> &distribution_processing,
        poisson_distribution<int> &distribution_repair,
        default_random_engine &generator)
    {
        status_changed = false;
        if (real_application)
            income_real++;
        if (!virus)
        {
            if (status != 0)
            {
                if (status == 1)
                {
                    time_real++;
                    int try_to_process = distribution_processing(generator);
                    if (try_to_process)
                    {
                        status = 0;
                        status_changed = true;
                    }
                }
                else
                {
                    time_virus++;
                    int try_to_repair = distribution_repair(generator);
                    if (try_to_repair)
                    {
                        status = 0;
                        status_changed = true;
                    }
                }
            }
            else if (virus_application > 0 || real_application > 0)
            {
                status_changed = true;
```

```
57                  if (!(virus_application > 0 && real_application > 0))
58                  {
59                      if (real_application > 0)
60                      {
61                          status = 1;
62                          real_app_done++;
63                          real_application--;
64                      }
65                      else
66                      {
67                          status = -1;
68                          virus_application--;
69                          virus = true;
70                      }
71                      return true;
72                  }
73                  else
74                  {
75                      int choise = rand() % 2;
76                      if (choise == 0)
77                      {
78                          status = -1;
79                          virus_application--;
80                          virus = true;
81                      }
82                      else
83                      {
84                          status = 1;
85                          real_app_done++;
86                          real_application--;
87                      }
88                      return true;
89                  }
90              }
91              else
92              {
93                  time_free++;
94                  return false;
95              }
96          }
97          else
98          {
99              time_virus++;
100             int try_to_process = distribution_processing(generator);
101             if (try_to_process)
102             {
103                 virus = false;
104             }
105             return false;
106         }
107     }
108     Channel()
109     {
110         time_real = 0;
111         time_free = 0;
112         time_virus = 0;
```

```
113         real_app_done = 0;
114         status = 0;
115         income_real = 0;
116     }
117     ~Channel() {};
118 private:
119 
120 };
121 class MaintenanceSystem
122 {
123 public:
124     vector<Channel> channels;
125     vector<vector<int> > status_history;
126     map<vector<int>, int> status_statistics;
127     int T;
128     int num_lost;
129     int num_accepted;
130     MaintenanceSystem(int num_of_channels)
131     {
132         channels.resize(num_of_channels, Channel());
133         num_lost = 0;
134         num_accepted = 0;
135     }
136     void run(double real_intens, double virus_intens, double
      processing_intens, double repair_intens, int _T)
137     {
138         T = _T;
139         vector<int> status(channels.size(), 0);
140         status_history.push_back(status);
141         int length = 0;
142         ofstream out("process.txt");
143         ofstream out_s("sequence.txt");
144         default_random_engine generator;
145         poisson_distribution<int> distribution_virus(virus_intens);
146         poisson_distribution<int> distribution_real(real_intens);
147         poisson_distribution<int> distribution_processing(processing_intens);
148         poisson_distribution<int> distribution_repair(repair_intens);
149         int count_real = 0, count_virus = 0;
150         for (int i = 0; i < T; i++)
151         {
152             int real = distribution_real(generator);
153             int virus = distribution_virus(generator);
154             count_real += real;
155             count_virus += virus;
156             out_s << real << ' ' << virus << endl;
157             length++;
158             bool status_changed = false;
159             bool accepted = false;
160             for (int c = 0; c < channels.size(); c++)
161             {
162                 bool  application_accepted = channels[c].input(real, virus,
```

```
                    distribution_processing, distribution_repair, generator);
163                 if (channels[c].status_changed)
164                 {
165                     status_changed = true;
166                     status[c] = channels[c].status;
167                 }
168             }
169             if (status_changed)
170             {
171                 for (int i = 0; i < status_history.back().size(); i++)
172                 {
173                     out.width(3);
174                     out << status_history.back()[i];
175                 }
176                 out.width(7);
177                 out << length << " it = ";
178                 out.width(11);
179                 out << double(length) / T * 100 << " %" << endl;
180                 status_statistics[status_history.back()] += length;
181                 status_history.push_back(status);
182                 length = 0;
183             }
184         }
185         status_statistics[status_history.back()] += length;
186         status_history.push_back(status);
187         length = 0;
188         out = ofstream("stats.txt");
189         out << endl << endl << "############## STATS ##############" << endl <<
              endl;
190         map <vector<int>, int>::iterator cur;
191         for (cur = status_statistics.begin();cur != status_statistics.end
              ();cur++)
192         {
193             for (int i = 0; i < (*cur).first.size(); i++)
194             {
195                 out.width(3);
196                 out << (*cur).first[i];
197             }
198             out.width(10);
199             out<< (*cur).second<< " it = ";
200             out.width(7);
201             out << double((*cur).second) / T * 100 << " %" << endl;
202         }
203         out << endl << "real applications for channel:" << endl;
204         double summ = 0;
205         for (int i = 0; i < channels.size(); i++)
206         {
207             out << "channel " << i + 1 << " = " << channels[i].real_app_done
                    << " = " << double(channels[i].real_app_done) / T << " per t"
                    <<"    income real = "<<channels[i].income_real<< endl;
208             summ += double(channels[i].real_app_done) / T;
209             num_accepted += channels[i].real_app_done;
210         }
211         num_lost = count_real - num_accepted;
212         out << "summ = " << summ << endl;
213         out << endl << "TOTAL REAL ACCEPTED = " << num_accepted << " = " <<
```

```
            double(num_accepted) / (num_lost + num_accepted) * 100 << " %" <<
            endl;
214         out << endl << "TOTAL REAL LOST = " << num_lost << " = " << double
            (num_lost) / (num_lost + num_accepted) * 100 << " %" << "\t" <<
            double(num_lost) / T << endl;
215     }
216     ~MaintenanceSystem() {}
217 private:
218
219 };
220
221 int main()
222 {
223     srand(time(NULL));
224     double lamda = 0.416, a = 0.93;
225     double virus_intens = (1 - a)*lamda, real_intens = a*lamda;
226     double processing_intens = 0.017, repair_intens = 0.05;
227     int T = 43200;
228     MaintenanceSystem sys(42);
229     sys.run(real_intens, virus_intens, processing_intens, repair_intens, T);
230 }
```